1. What will be the output of the below code

package com.abc.lambdas;

import java.util.Arrays;

import java.util.List;

interface Consumer {

void execute(Integer element);

}

public class Functional1 {

private static void forEach(List<Integer> elements, Consumer consumer) {

for (Integer no : elements) {

consumer.execute(no);

}

}

public static void main(String[] args) {

List<Integer> nos = Arrays.asList(5, 6, 2, 3, 8, 9, 7, 10, 1, 3);

forEach(nos, new Consumer() {

@Override

public void execute(Integer element) {

if (element % 2 != 0) {

System.out.print(element + " ");

}

}

});

}

}

1. Will not compile
2. Will give an error at run time
3. 6 2 8 10
4. 5 3 9 7 1 3

2. What will be the output of the below code

import java.util.Arrays;

import java.util.List;

public class Functional2 {

public static void main(String[] args) {

List<Integer> nos = Arrays.asList(5, 6, 2, 3, 8, 9, 7, 10, 1, 3);

nos.forEach((element) -> {

if (element % 2 == 0 && element > 2) {

System.out.println(element);

}

});

}

}

1. 6 2 8 10
2. 5 3 9 7 1 3
3. Will give compilation error
4. 6 8 10

3. What will be the output of the below code:

import java.util.Arrays;

import java.util.List;

interface Consumer {

void execute(Integer element);

void func(Float element);

}

class Functional2 {

private static void forEach(List<Integer> elements, Consumer consumer) {

for (Integer no : elements) {

consumer.execute(no);

}

}

public static void main(String[] args) {

List<Integer> nos = Arrays.asList(5, 6, 2, 3, 8, 9, 7, 10, 1, 3);

forEach(nos, element -> {

if (element % 2 == 0) {

System.out.print(element + " ");

}

});

}

}

1. 6 2 8 10
2. Compilation error
3. Runtime error
4. 5 3 9 7 1 3

4. It is compulsory for `single abstract method` interfaces to be marked with the @FunctionalInterface annotation. Is this statement true ?

1. Yes
2. No

5. Is the below Spring component valid ? Assume the StudentServiceIntf interface already exists and is valid.

package com.abc.libmgmt.service;

import java.util.Optional;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.data.domain.Example;

import org.springframework.stereotype.Service;

import com.abc.libmgmt.dao.StudentDaoIntf;

import com.abc.libmgmt.domain.Student;

@Service

public class StudentServiceImpl implements StudentServiceIntf {

private StudentDaoIntf studentDao;

@Override

public Student createNewStudent(Student student) {

return studentDao.save(student);

}

@Override

public Optional<Student> exists(Student student) {

return studentDao.findOne(Example.of(student));

}

}

1. Yes
2. No

6. What will the HTTP response code sent back by the `createBook()`

package com.example.firstapp.controllers;

import java.util.ArrayList;

import com.example.firstapp.domain.Book;

import com.example.firstapp.services.BookService;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.http.HttpStatus;

import org.springframework.web.bind.annotation.GetMapping;

import org.springframework.web.bind.annotation.PostMapping;

import org.springframework.web.bind.annotation.RequestBody;

import org.springframework.web.bind.annotation.ResponseStatus;

import org.springframework.web.bind.annotation.RestController;

import org.springframework.web.server.ResponseStatusException;

@RestController

class BookController {

@Autowired

private BookService bookService;

@GetMapping("/books")

public ArrayList<Book> getBooks() {

return bookService.getAllBooks();

}

@PostMapping("/books")

public Book createBook(@RequestBody Book book) {

try {

return bookService.saveBook(book);

} catch (Exception e) {

throw new ResponseStatusException(HttpStatus.BAD\_REQUEST, e.getMessage(), e);

}

}

}

1. 201
2. 200
3. 204
4. None of the above

7. The below JPA entity and JpaRepository interface can be used to retrieve all the records from the database.

package com.example.firstapp.domain;

import javax.persistence.Entity;

import javax.persistence.GeneratedValue;

import javax.persistence.GenerationType;

import javax.persistence.Id;

@Entity

public class Book {

@Id

@GeneratedValue(strategy = GenerationType.AUTO)

private Long id;

private String title;

private Integer pages;

private Float price;

public Book(Long id, String title, Integer pages, Float price) {

this.id = id;

this.title = title;

this.pages = pages;

this.price = price;

}

public Integer getPages() {

return pages;

}

public void setPages(Integer pages) {

this.pages = pages;

}

public Float getPrice() {

return price;

}

public void setPrice(Float price) {

this.price = price;

}

public String getTitle() {

return title;

}

public void setTitle(String title) {

this.title = title;

}

public Long getId() {

return id;

}

public void setId(Long id) {

this.id = id;

}

}

1. Yes, the statement is true
2. No, it is not true as there is an error in the above JPA entity class

8. In order to connect with a mysql database using Spring data jpa, are all the dependencies listed here, complete ?

<dependencies>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-web</artifactId>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-data-jpa</artifactId>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-test</artifactId>

<scope>test</scope>

<exclusions>

<exclusion>

<groupId>org.junit.vintage</groupId>

<artifactId>junit-vintage-engine</artifactId>

</exclusion>

</exclusions>

</dependency>

</dependencies>

1. Yes
2. No, there are some dependencies that are absent